Finding Invalid Signatures in Pairing-based Batches†

Laurie Law
National Security Agency

ECC 2006

(Based on joint work with Brian Matt, SPARTA, Inc.) ‡

†The views and conclusions contained in this presentation are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Security Agency, the Army Research Laboratory, or the U. S. Government.

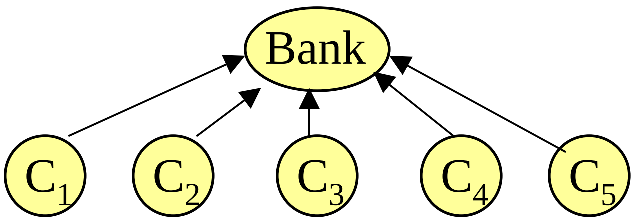
‡Dr. Matt's work through collaborative participation in the Communications and Networks Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD-19-01-2-0011.

Batch Verification of Digital Signatures

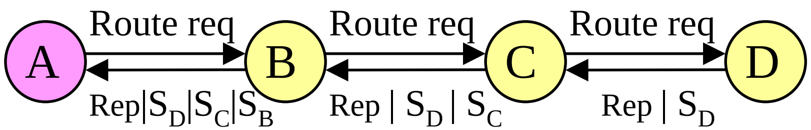
- A digital signature authenticates the source of a message and that the message has not been altered
  - Message is signed with signer’s private key
  - Signer’s public key is used to verify signature
- If most signatures are valid, can save time by verifying a “batch” of signatures together
  - What is the fastest way to verify the batch?
  - If the batch fails, how to quickly identify the bad signatures?

Applications

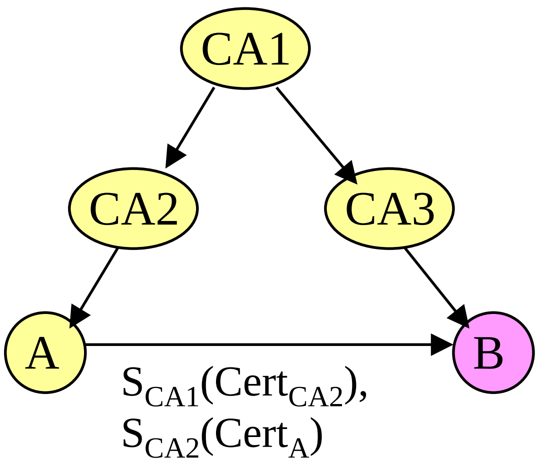
Check processing



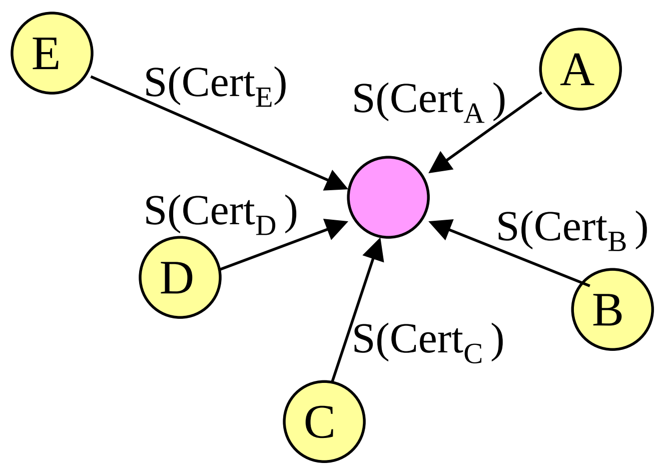
Routing security



Validating PKI
Certificate chains



Authenticating neighboring nodes



Outline

- Background
- Faster identification of invalid signatures
- New techniques for pairing-based signatures
- Cost comparisons

Background

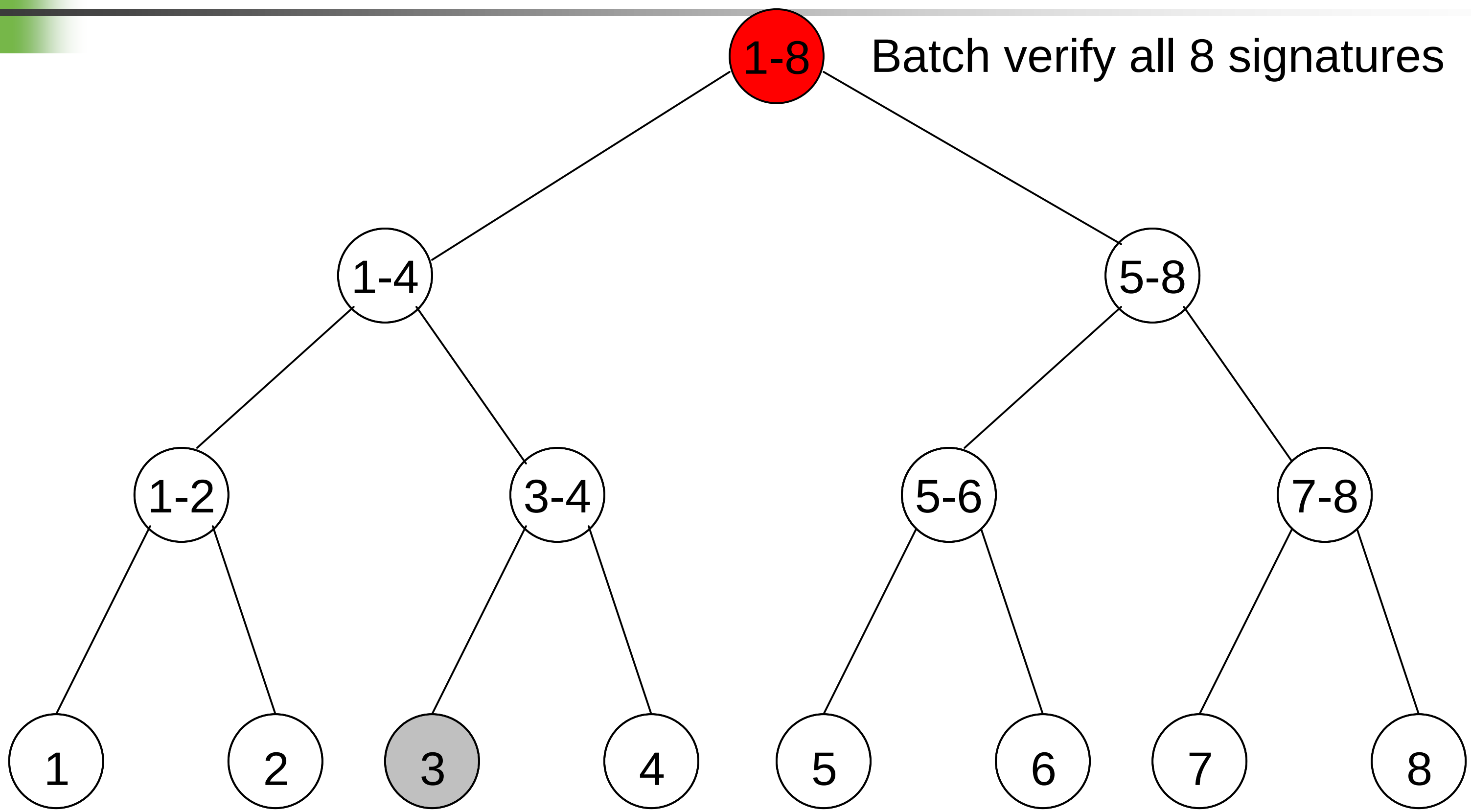
Batch Verification

- G is a prime order group
- $x_i \in Z_p$, $y_i \in G$, g is a generator of G
- Given (x_1, y_1) , (x_2, y_2) , … , (x_N, y_N)
  - Need to verify that $g^{x_i} = y_i$ for all $i=1$ to N
- Small exponents test (Bellare et al. 1998)
  - Pick small random m-bit integers $r_1, r_2, \ldots, r_N$
  - Compute $x = \sum r_i x_i$, $y = \prod y_i^{r_i}$
  - If $g^x = y$ then accept; otherwise reject
- The probability that test accepts a bad batch is at most 2^{-m}

Identifying bad signatures

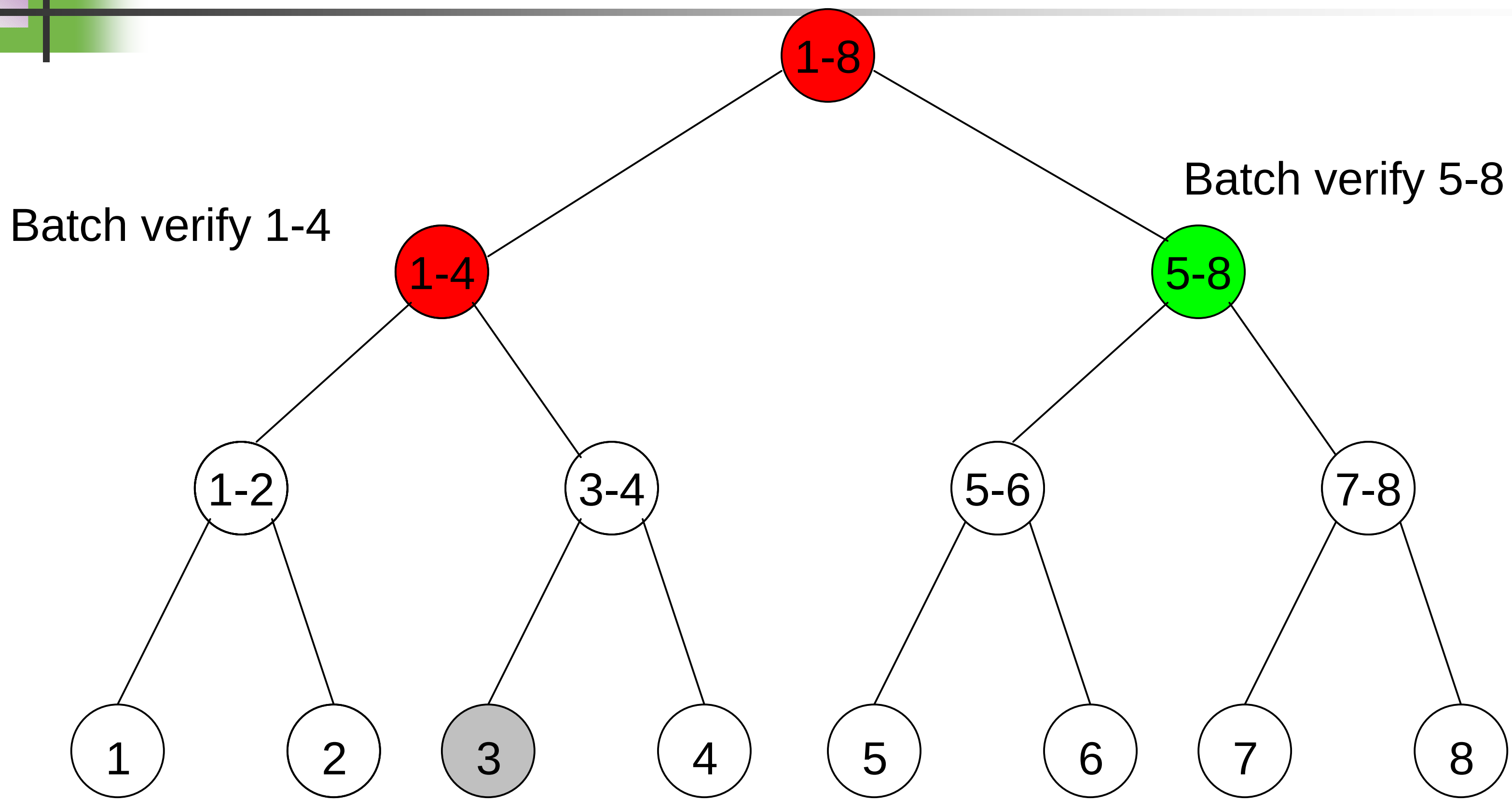
- Verify each signature individually
- Divide and conquer
  - Pastuzak et al. (PKC 2000)
  - Recursively divide into sub-batches
- Applications to RSA signatures
  - Lee, Cho, Choi, Cho 2006
  - Problem found with this approach to batch RSA (Stanek 2006)

Divide and Conquer: Simple Binary Search



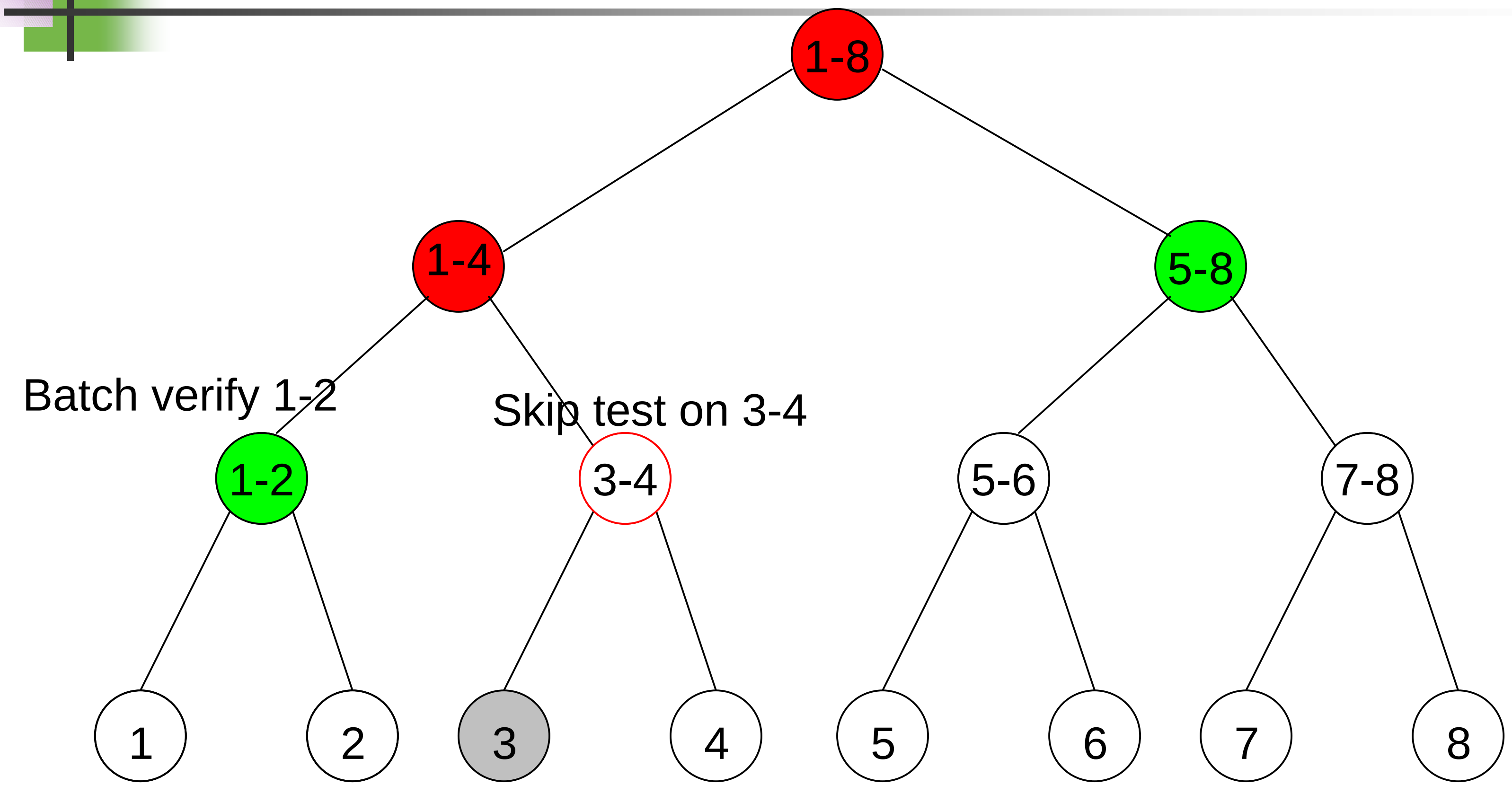
Signature 3 is invalid

Simple Binary Search



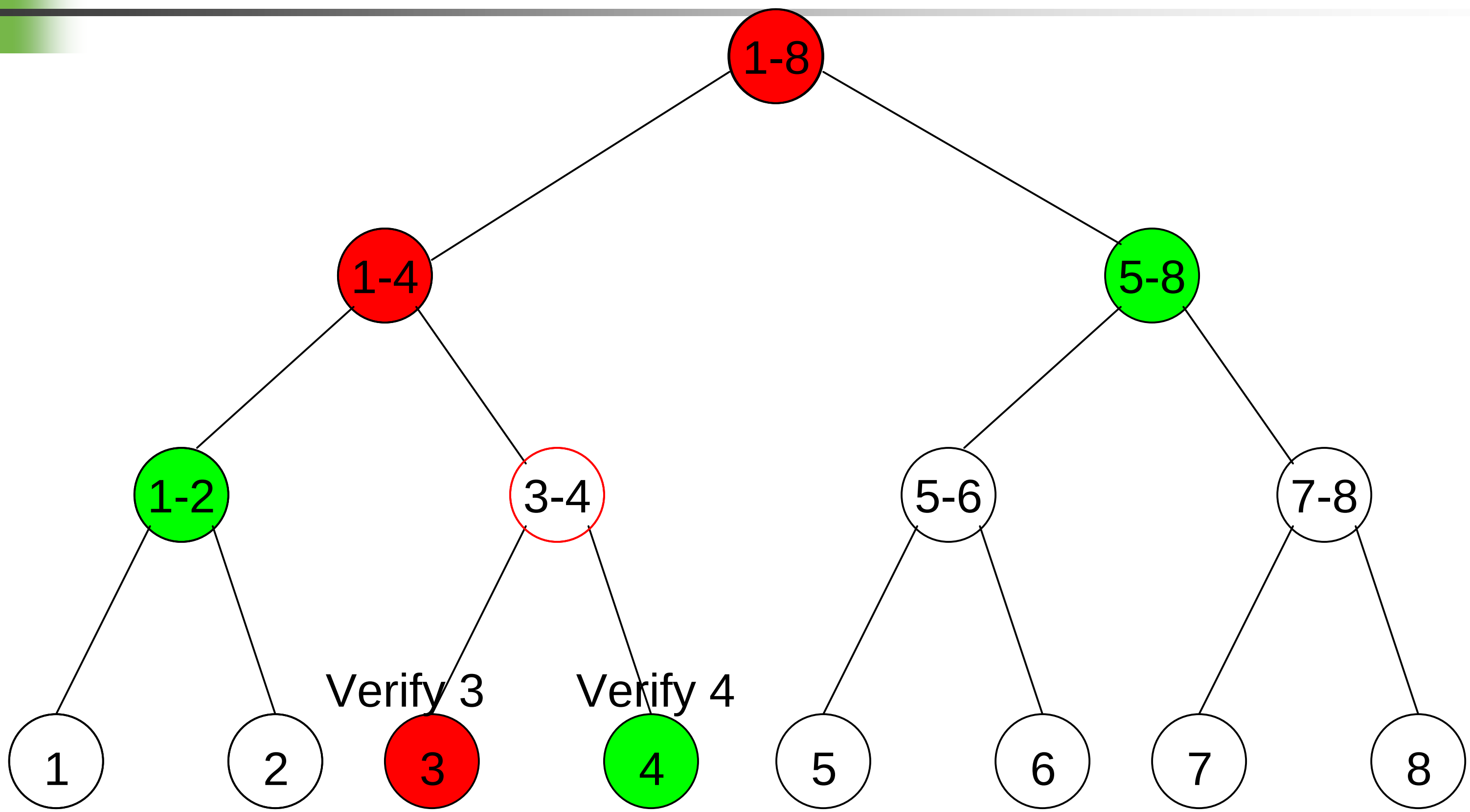
Signature 3 is invalid

Simple Binary Search



Signature 3 is invalid

Simple Binary Search



Signature 3 is invalid
5 verifications (beyond initial)
Maximum # verifications for N signatures (1 invalid): 2 lg(N)

Faster identification of invalid signatures

Improvement to Simple Binary Search

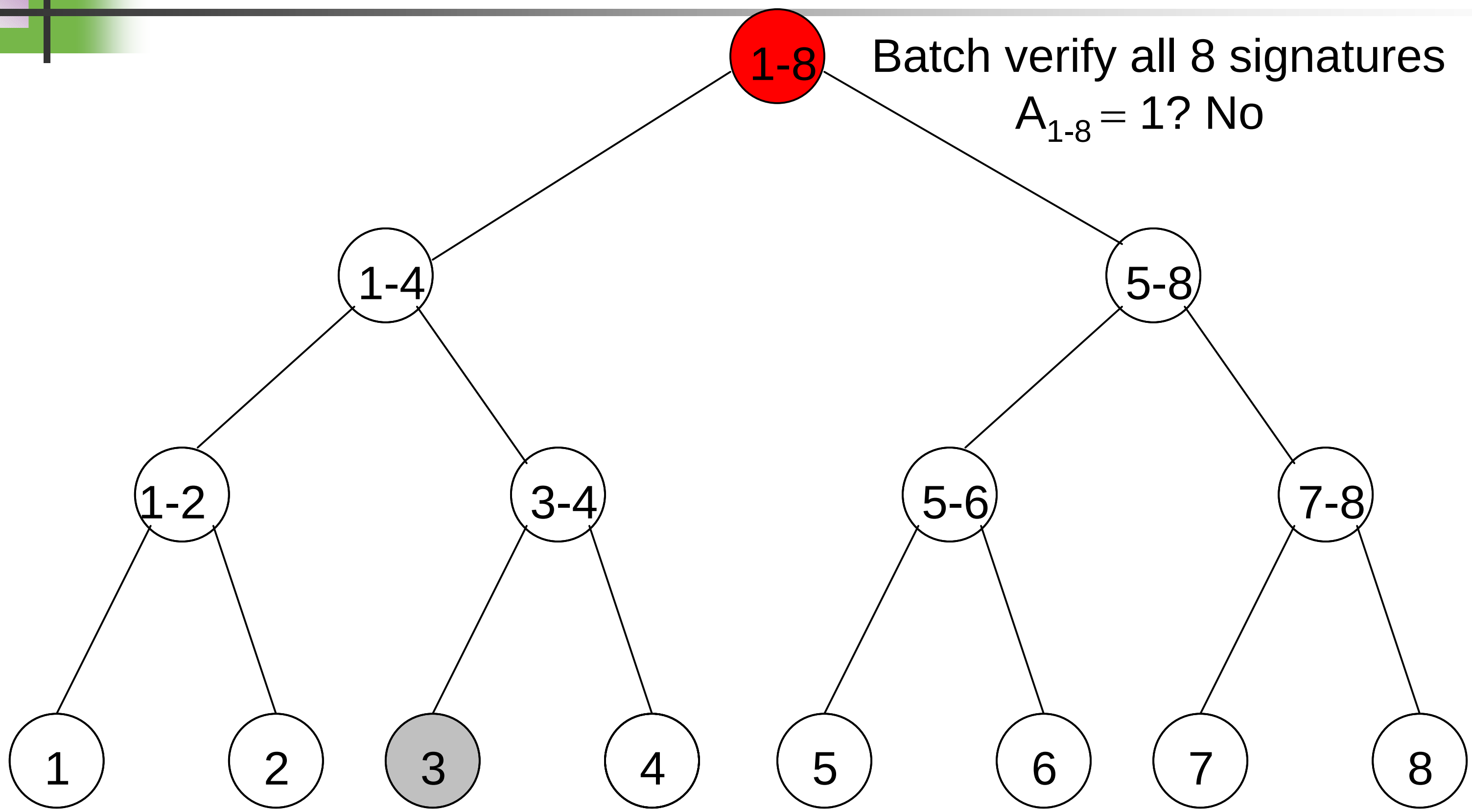
- Batch verification typically asks "Is $X=Y$?"
- Instead, compute $A=XY^{-1}$
  - $A=1 \Leftrightarrow$ batch is valid
- For batch of signatures (X_i, Y_i), $i=1$ *to* N

$$A = \prod_{i=1}^{N} A_i = A_{S_1} * A_{S_2}$$

$$A_{S_1} = \left(\prod_{i \in S_1} A_i \right), A_{S_2} = \left(\prod_{i \in S_2} A_i \right)$$

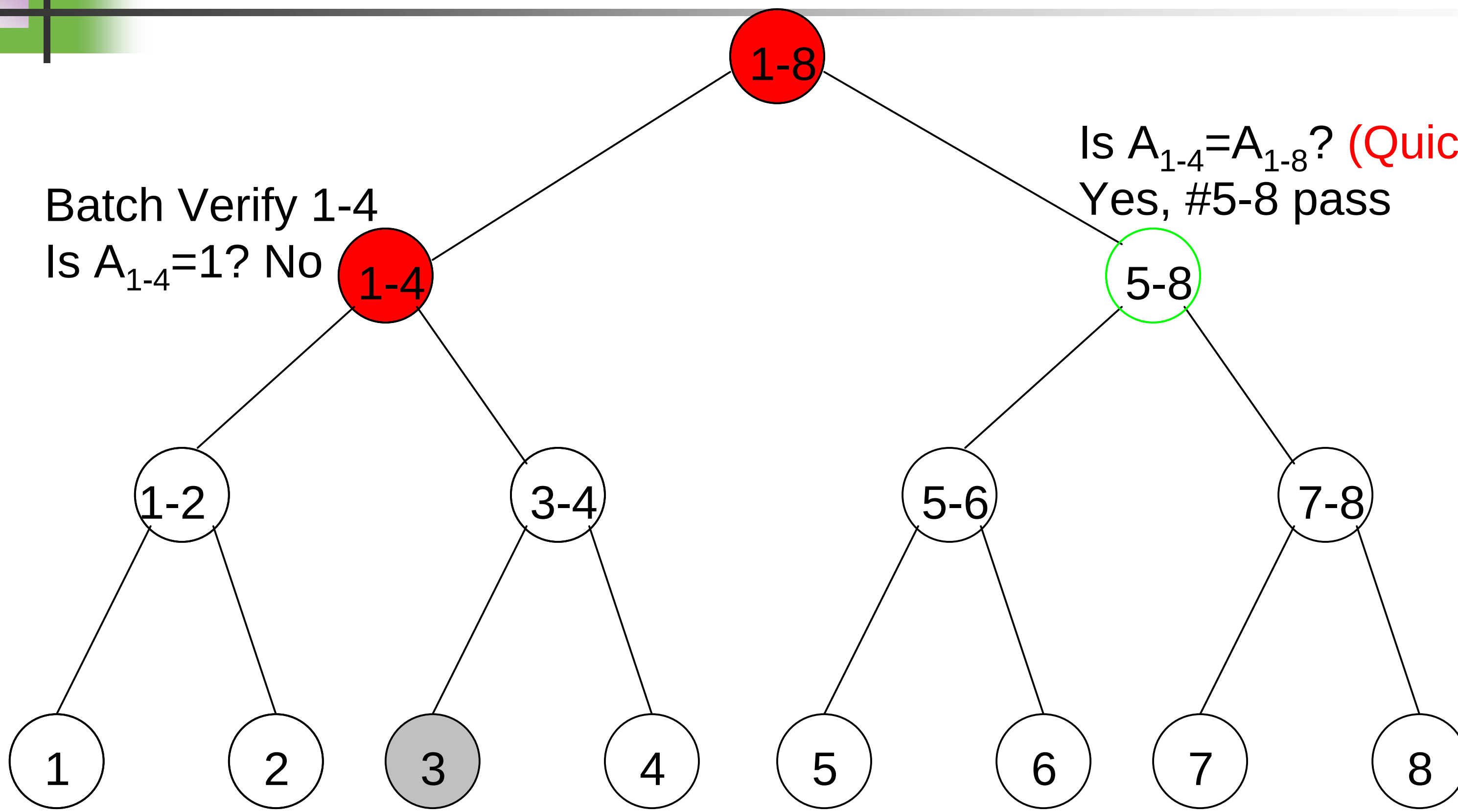
- $A \neq 1$ and $A_{S_1} = 1$ → $A_{S_2} \neq 1$, S_2 bad (skip verify)
- $A \neq 1$ and $A_{S_1} \neq 1$ → now do "Quick Test" on S_2
  - $A = A_{S_1}$ → $A_{S_2} = 1$, S_2 is good
  - $A \neq A_{S_1}$ → $A_{S_2} \neq 1$, S_2 is bad

Quick Binary Search



Signature 3 is invalid

Quick Binary Search

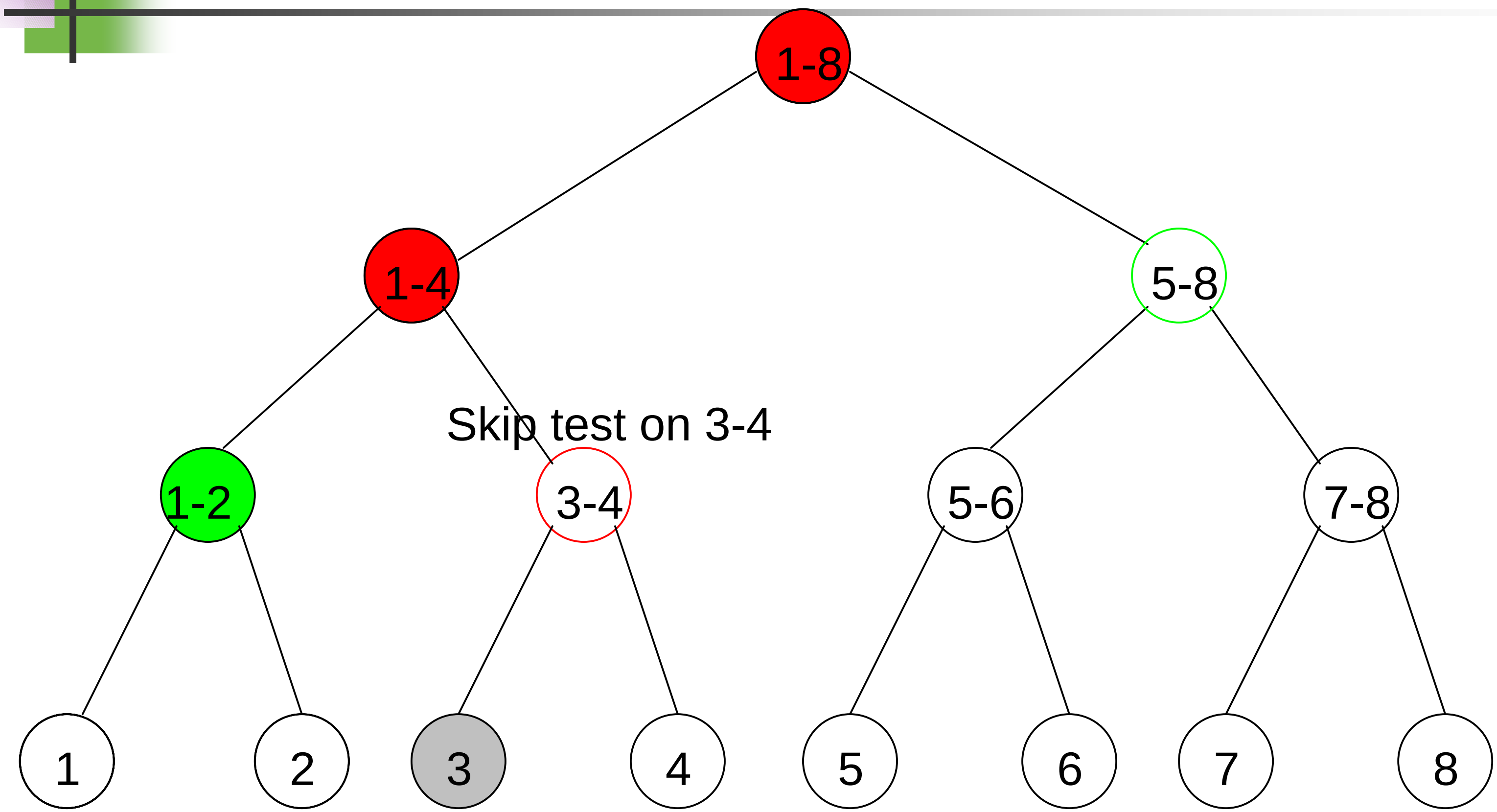


Is $A_{1\text{-}4}=A_{1\text{-}8}$? (Quick test)
Yes, #5-8 pass

Batch Verify 1-4
Is $A_{1\text{-}4}=1$? No

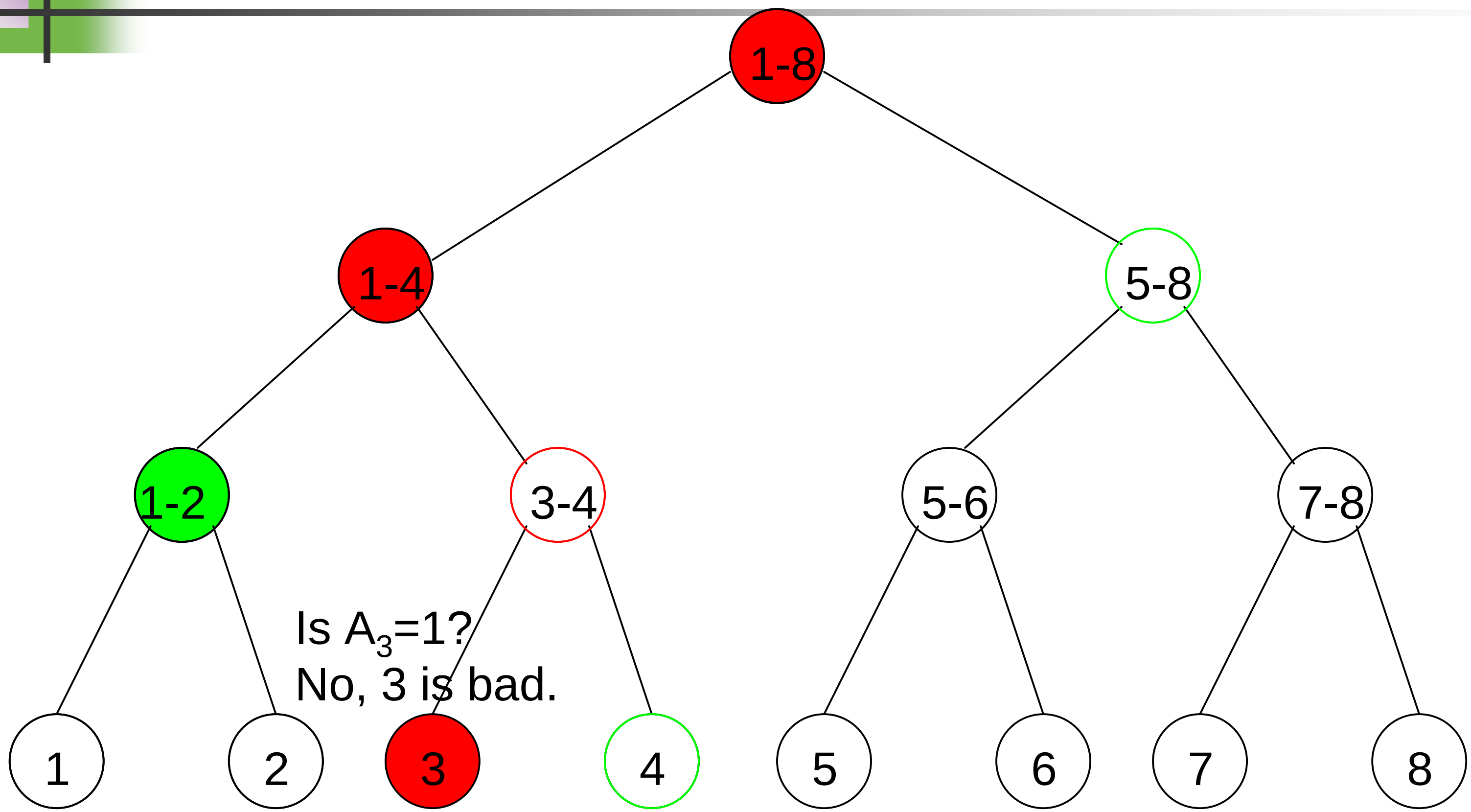
Signature 3 is invalid

Quick Binary Search



Signature 3 is invalid

Quick Binary Search



A_4=$A_{3\text{-}4}$? (Quick test)
Yes, 4 is good.

3 verifications (beyond initial)
verifications for *N* signatures (1 invalid): lg(*N*)

Cost (# verifications - worst case)

- 1 invalid signature
  - Simple Binary: $2\lceil lg\, N \rceil$
  - Quick Binary: $\lceil lg\, N \rceil$
- w bad signatures
  - Simple Binary:

    $$2(2^{\lceil lg\, w \rceil}-1+w(\lceil lg\, N \rceil - \lceil lg\, w \rceil))$$
  - Quick Binary:

    $$2^{\lceil lg\, w \rceil}-1+w(\lceil lg\, N \rceil - \lceil lg\, w \rceil)$$

New techniques for pairing-based signatures

Pairing-based Signatures

- Pairings have been used in identity-based and short signatures
- Identity-based: public key can be easily derived from identity so certificates are not needed
- Very efficient in wireless networks

Sender ID	Message	Signature	Sender's Public Key	Certificate	(cert chain)

Not needed!

- Drawback – verification of many schemes requires 2 expensive bilinear pairings per signature

Bilinear pairings on elliptic curves

- E is an elliptic curve defined over F_q, q prime
- r is a prime divisor of $\#E(F_q)$
- Q and R are points of order r
- $<Q, R>$ maps Q and R into order r subgroup of F_{q^d}

$$<Q, R_0+R_1> = <Q, R_0><Q, R_1>$$
$$<Q_0+Q_1, R> = <Q_0, R><Q_1, R>$$
$$<kQ, R> = <Q, kR> = <Q, R>^k$$

Cha-Cheon signature (2003)

- System set-up

  s = master key (secret integer)

  R = order r point on $E(F_{q^d})$ - $E(F_q)$ (public)

  $P = sR$ (public)

- Signer's key pair

  Public: Q is an order r point on $E(F_q)$

  Private: $D = sQ$

- Signing a message m:

  $U = tQ$ (t randomly generated by signer)

  $V = (t + \text{hash}(m,U))D$

- Verification:

  Accept if received points are in the correct group and

  $$<U + \text{hash}(m,U)Q, P> = <V, R>$$

Batch Verification for Cha-Cheon

- Apply small exponents test
- For k = 1 to N, the verifier receives
  - m_k: message
  - Q_k: signer's public key
  - U_k, V_k: signature of m_k
- Verifier validates received points and generates random integers $r_1 = 1$, r_2, ... , r_N

$$B_k = r_k\left(U_k + hash\left(m, U_k\right)Q_k\right)$$

$$D_k = r_k V_k$$

- Batch is valid $\Leftrightarrow$ $\left\langle \sum_{k=1}^{N} B_k, P \right\rangle = \left\langle \sum_{k=1}^{N} D_k, R \right\rangle$

Finding the invalid signatures

- Quick Binary Search
  - Rewrite initial verification:

$$A_0 = \left\langle \sum_{k=1}^{N} B_k, P \right\rangle \left\langle \sum_{k=1}^{N} D_k, -R \right\rangle$$

- A_0=1 → batch is valid
- Finding 1 bad signature requires $2\lg N$ pairings
- Can we reduce the number of pairings (for a small # of bad signatures)?

Exponentiation Method

- If initial verification fails, compute

$$A_1 = \left\langle \sum_{k=1}^{N} kB_k, P \right\rangle \left\langle \sum_{k=1}^{N} kD_k, -R \right\rangle$$

- If i is the only invalid signature, then

$$A_1 = \prod_{k=1}^{N} \langle B_k, P\rangle^k \langle D_k, -R\rangle^k = \left(\langle B_i, P\rangle\langle D_i, -R\rangle\right)^i = A_0^i$$

- If $A_1 = A_0{}^i$ then the i^{th} signature is invalid
- No match → at least 2 bad signatures

Identifying 2 bad signatures

- Compute

$$A_2 = \left\langle \sum_{k=1}^{N} k\left(kB_k\right), P \right\rangle \left\langle \sum_{k=1}^{N} k\left(kD_k\right), -R \right\rangle$$

- Find $i, j \in [1, N]$, $i < j$ such that

$$A_2 = A_1^{i+j} A_0^{-ij}$$

- Signatures i and j are invalid
- No match → at least 3 bad signatures

Identifying w bad signatures

- Compute

$$A_w = \left\langle \sum_{k=1}^{N} k\left(k^{w-1} B_k\right), P \right\rangle \left\langle \sum_{k=1}^{N} k\left(k^{w-1} D_k\right), -R \right\rangle$$

- Find $x_1, \ldots, x_w \in [1, N]$, $x_1 < \ldots < x_w$ such that

$$A_w = \prod_{t=1}^{w} \left(A_{w-t}^{(-1)^{t-1}} \right)^{p_t} \qquad (1)$$

where p_t is the t^{th} elementary symmetric polynomial in $x_1, \ldots, x_w$

- Signatures $x_1, \ldots, x_w$ are invalid
- No match → at least $w+1$ bad signatures

Costs for Exponentiation Method (To test for w bad signatures)

- Compute A_1 through A_w
  - $2w$ pairings
  - $2w$ (N-1) short elliptic scalar multiplies
    - Can be implemented with $2w(N$-1)EC additions
  - w multiplies in F_{q^d}
- Find w-tuple (x_0, x_1, ..., x_w) to solve (1)
  - w-1 inverses in F_{q^d}
  - To test all w-tuples: approx w(N choose w) < N^w multiplies in F_{q^d}
    - Square-root discrete log methods are faster for small w

Using discrete log methods to find invalid signatures

- To find a single bad signature, find $i \in [1, N]$ such that $A_1 = A_0^{\,i}$
- Using Shanks' "baby-step giant-step":

$$i = c + d\sqrt{N}$$

$$1 \le c, d \le \sqrt{N}$$

$$A_1 A_0^{-c} = A_0^{d\sqrt{N}}$$

- $2N^{1/2}$ multiplies in F_{q^d}

Baby Step-Giant Step (2 invalid signatures)

- Find $p_1 = i+j$ and $p_2 = ij$ such that

$$A_2 = A_1^{p_1} A_0^{-p_2}$$

$$1 \le p_1 \le 2N, 1 \le p_2 \le N^2$$

$$p_1 = c_1 + d_1\sqrt{2N},\ p_2 = c_2 + d_2 N$$

$$1 \le c_1, d_1 \le \sqrt{2N}, 1 \le c_2, d_2 \le N$$

$$A_2 A_1^{-c_1} A_0^{c_2} = A_1^{d_1\sqrt{2N}} A_0^{-d_2 N}$$

- $(2N)^{3/2}$ multiplies to find p_1 and p_2

Baby-Step Giant-Step (generalized)

- For w invalid signatures, the number of multiplies are:

$$2\left(\prod_{i=1}^{w}\binom{w}{i}\right)^{1/2} N^{w(w+1)/4}$$

- This is faster than testing all w-tuples when $w<3$

w	# multiplies
1	$2N^{1/2}$
2	$(2N)^{3/2}$
3	$6N^3$

Exponentiation with sectors

- Divide N signatures into S sectors of N/S signatures
- **Stage 1**: Find the bad sectors using the exponentiation method but with multipliers equal to the sector ID

1	1	1	1	2	2	2	2	3	3	3	3	4	4	4	4

- **Stage 2**: Find bad signatures using the original exponentiation method (can reuse A_i's from previous tests) but test only signatures from bad sectors

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Cost comparisons

Approximate cost to identify w bad signatures in a failed batch of N signatures

Method	Pairings	Inverses in F_{q^d}	EC additions	Multiplies in F_{q^d}
Simple Binary (worst case)	$4w \lg N$	0	0	0
Quick Binary (worst case)	$2w \lg N$	0	0	0
Exponentiation	$2w$	w-1	$2w(N - 1)$	$\min(N^w, f_w N^{w(w+1)/4})$
Exponentiation with S Sectors*	$4w$	$1.5(w-1)$	$4w(N - 1)$	$<2 f_w N^{w(w+1)/8}$

* Assumes 1 bad signature per sector and $S = N^{1/2}$.

Costs

- Parameter sizes
  - $|r|$ = 160 bits
  - $|q| \cong$ 160 bits (signature length = 2*$|q|$)
  - d = 6 (embedding degree)
- Estimates for relative costs of operations (from Granger, Page and Smart, ANTS 2006)
  - 1 pairing = 9120 multiplies in F_q
  - 1 multiply in F_{q^6} = 15 multiplies in F_q
  - 1 inverse in F_{q^6} = 274 multiplies in F_q
  - 1 EC addition = 11 multiplies in F_q

Cost to find 1 invalid signature (# multiplies in F_q)

N	Simple Binary	Quick Binary	Exp	$N^{1/2}$ Sectors
10	145920	72960	18558	36996
100	255360	127680	20718	41076
1000	364800	182400	41178	80796
10000	510720	255360	241218	477036
100000	620160	310080	2227728	4437516

Cost to find 2 invalid signatures (# multiplies in F_q)

N	Simple Binary	Quick Binary	Exp	$N^{1/2}$ Sectors
10	255360	127680	38650	74780*
100	474240	237120	86110	84905
1000	693120	346560	1430710	176510
10000	984960	492480	43076710	1038275
100000	1203840	601920	1348436710	9350585

*Will be faster if both signatures fall in the same sector.

Cost to find 3 invalid signatures (# multiplies in F_q)

N	Simple Binary	Quick Binary	Exp	$N^{1/2}$ Sectors
10	328320	164160	63362	116861*
100	656640	328320	7561802	303056
1000	984960	492480	$7.5*10^9$	5933951
10000	1422720	711360	$7.5*10^{12}$	$1.8*10^8$
100000	1751040	875520	$7.5*10^{15}$	$5.7*10^9$

*Will be faster if some invalid signatures fall in the same sector.

Conclusions

- New methods for finding invalid signatures in failed batches
  - Improved general method
  - Other methods for pairing-based schemes with small to medium-sized batches
  - One or more of these methods will beat earlier techniques if # invalid signatures is small
  - Combine methods for optimal results